lh

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/990,569	11/21/2001	Aaron S. Mar	004906.P055	2639

8791 7590 05/02/2005

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
MARTIN, CIARA A	
ART UNIT	**PAPER NUMBER**
2157	

DATE MAILED: 05/02/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

K

Office Action Summary	Application No.	Applicant(s)
	09/990,569	MAR, AARON S.
	Examiner	**Art Unit**
	Ciara Martin	2157

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on *07 April 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-44* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-44* is/are rejected.

7)☒ Claim(s) *2, 5, 15, 17, 24, 27, 37 and 39* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

K

DETAILED ACTION

1. This action is responsive to the amendment filed on April 7, 2005. Claims 1-4, 8, 10, 12-26, 30, 32 and 34-44 were amended. Claims 1-44 are pending. Claims 1-44 represent policy change characterization method and apparatus.

Claim Objections

2. Claims 2 and 24 are objected to because the word "the" is missing from the phrase "...wherein each of packets includes...".

3. Claims 15 and 37 are objected to because the word "hope" in the phrase "...a next hope of the data packets..." should be the word "hop".

4. Claims 17 and 39 are objected to because of the following informalities: the word "the" is missing from the phrase "...perform following operations:" Add the word "the" between "perform" and "following". The word "baed" in the phrase "next hop baed on a class ID" should be "based".

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1, 23, and 44 are rejected under 35 U.S.C. 102(b) as being anticipated by Simmonds et al. U.S. Patent No. 5,893,116.

7. Simmonds teaches the invention as claimed including a method and machine for replicating network resources that are arranged as a hierarchical tree (see abstract and fig. 1).

8. As per claim 1, Simmonds teaches a method comprising:

a. receiving a new policy tree at a network element in a network, wherein the network element stores a current policy tree of classes for quality of service (QoS) of packets being processed by the network element (see c. 6, ll. 57-65 and c. 9, ll. 10-13; Simmonds discloses a mobile server which stores replicated resources in a network, the network resources are arranged in a directory subtree; the replicated resources are the current tree, the network resources are the new tree; a mobile server is a network element, and a subtree is a tree); and wherein the classes of the current policy tree and the classes of the new policy tree comprise leaf classes and non-leaf classes (see c. 9, ll. 45-55; Simmonds discloses the replication database stores information container objects and leaf objects; an object is a class, the replication database is the current tree and container objects are non-leaf classes).

b. comparing the classes of the current policy tree with the classes of the new policy tree (see c. 7, l. 66 to c. 8 ll. 1-5; Simmonds discloses replicating and propagating changes made on the network resource tree to the replication database; the replication database is the current tree, and replicating and propagating changes involves first comparing); including

i. for the current policy tree and the new policy tree, merging, into a set of classification rules of the leaf classes, classification rules of non-leaf classes that are ancestors of the leaf classes (see c. 9, ll. 60-67; Simmonds discloses leaf objects inherit inheritance rules from the container; inheritance rules are classification rules and containers are non-leaf classes).

ii. identifying a leaf class in the current policy tree as identical to a leaf class in the new policy tree upon determining that the set of classification rules of the leaf class in the current policy tree is equal to the set of classification rules of the leaf class in the new policy tree (see c. 12, ll. 39-52; Simmonds discloses synchronizing leaf objects from a replica if a change has occurred; it is inherent that if a change has not occurred then there is no synchronization and the leaf objects are identical).

iii. identifying a non-leaf class in the current policy tree as identical to a non-leaf class in the new policy tree upon determining that the non-leaf class in the current policy tree and the non-leaf class in the new policy tree share a greatest number of equivalent descendant leaf classes (see c. 12, ll. 39-52; Simmonds discloses synchronizing resources from a replica if a change has occurred; it is inherent that if a change has not occurred then there is no synchronization and the resources are identical; a non-leaf object is a resource); and

iv. marking the classes of the current policy tree and the new policy tree as added, deleted, modified or unchanged based on the identifying of the identical leaf and non-leaf classes in the current policy tree and new policy tree (see c. 12, ll. 39-58 and table 1 in column 13; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include unchanged, changed [or modified] and deleted; changes are marked on the resources [classes]); and

c. selectively deleting classes of the current policy tree based on the comparison of the classes (see c. 7, l. 66 to c. 8 ll. 1-5 and c. 10, ll. 40-49; Simmonds discloses replicating and propagating changes made on the network resource tree to the replication database and selecting objects to delete based on the comparison; the replication database is the current tree, and replicating and propagating changes involves first comparing).

9. As per claim 23, Simmonds teaches a machine-readable medium that provides instructions, which when executed by a machine, cause said machine to perform operations comprising:

a. receiving a new policy tree at a network element in a network, wherein the network element stores a current policy tree of classes for quality of service (QoS) of packets being processed by the network element (see c. 6, ll. 57-65 and c. 9, ll. 10-13; Simmonds discloses a mobile server which stores replicated resources in a network, the network resources are arranged in a directory

subtree; the replicated resources are the current tree, the network resources are the new tree; a mobile server is a network element, and a subtree is a tree); and wherein the classes of the current policy tree and the classes of the new policy tree comprise leaf classes and non-leaf classes (see c. 9, ll. 45-55; Simmonds discloses the replication database stores information container objects and leaf objects; an object is a class, the replication database is the current tree and container objects are non-leaf classes).

b. comparing the classes of the current policy tree with the classes of the new policy tree (see c. 7, l. 66 to c. 8 ll. 1-5; Simmonds discloses replicating and propagating changes made on the network resource tree to the replication database; the replication database is the current tree, and replicating and propagating changes involves first comparing); including

i. for the current policy tree and the new policy tree, merging, into a set of classification rules of the leaf classes, classification rules of non-leaf classes that are ancestors of the leaf classes (see c. 9, ll. 60-67; Simmonds discloses leaf objects inherit inheritance rules from the container; inheritance rules are classification rules and containers are non-leaf classes).

ii. identifying a leaf class in the current policy tree as identical to a leaf class in the new policy tree upon determining that the set of classification rules of the leaf class in the current policy tree is equal to the set of classification rules of the leaf class in the new policy tree (see c. 12, ll. 39-

52; Simmonds discloses synchronizing leaf objects from a replica if a change has occurred; it is inherent that if a change has not occurred then there is no synchronization and the leaf objects are identical).

iii. identifying a non-leaf class in the current policy tree as identical to a non-leaf class in the new policy tree upon determining that the non-leaf class in the current policy tree and the non-leaf class in the new policy tree share a greatest number of equivalent descendant leaf classes (see c. 12, ll. 39-52; Simmonds discloses synchronizing resources from a replica if a change has occurred; it is inherent that if a change has not occurred then there is no synchronization and the resources are identical; a non-leaf object is a resource); and

iv. marking the classes of the current policy tree and the new policy tree as added, deleted, modified or unchanged based on the identifying of the identical leaf and non-leaf classes in the current policy tree and new policy tree (see c. 12, ll. 39-58 and table 1 in column 13; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include unchanged, changed [or modified] and deleted; changes are marked on the resources [classes]); and

c. selectively deleting classes of the current policy tree based on the comparison of the classes (see c. 7, l. 66 to c. 8 ll. 1-5 and c. 10, ll. 40-49; Simmonds discloses replicating and propagating changes made on the network

resource tree to the replication database and selecting objects to delete based on the comparison; the replication database is the current tree, and replicating and propagating changes involves first comparing).

10. As per claim 44, Simmonds teaches a network element comprising:

a. a processor (see c. 15, ll. 8-10; Simmonds discloses a computer, a computer is a processor); and

b. a memory coupled to the processor for storing instructions, when executed from the memory (see c. 15, ll. 11-16; Simmonds discloses a storing means for storing executed software, memory is storage), cause the processor to perform operations including:

i. receiving a new policy tree at a network element in a network, wherein the network element stores a current policy tree of classes for quality of service (QoS) of packets being processed by the network element (see c. 6, ll. 57-65 and c. 9, ll. 10-13; Simmonds discloses a mobile server which stores replicated resources in a network, the network resources are arranged in a directory subtree; the replicated resources are the current tree, the network resources are the new tree; a mobile server is a network element, and a subtree is a tree); and wherein the classes of the current policy tree and the classes of the new policy tree comprise leaf classes and non-leaf classes (see c. 9, ll. 45-55; Simmonds discloses the replication database stores information container objects and leaf objects;

an object is a class, the replication database is the current tree and container objects are non-leaf classes).

ii. comparing the classes of the current policy tree with the classes of the new policy tree (see c. 7, l. 66 to c. 8 ll. 1-5; Simmonds discloses replicating and propagating changes made on the network resource tree to the replication database; the replication database is the current tree, and replicating and propagating changes involves first comparing); including:

(1) for the current policy tree and the new policy tree, merging, into a set of classification rules of the leaf classes, classification rules of non-leaf classes that are ancestors of the leaf classes (see c. 9, ll. 60-67; Simmonds discloses leaf objects inherit inheritance rules from the container; inheritance rules are classification rules and containers are non-leaf classes).

(2) identifying a leaf class in the current policy tree as identical to a leaf class in the new policy tree upon determining that the set of classification rules of the leaf class in the current policy tree is equal to the set of classification rules of the leaf class in the new policy tree (see c. 12, ll. 39-52; Simmonds discloses synchronizing leaf objects from a replica if a change has occurred; it is inherent that if a change has not occurred then there is no synchronization and the leaf objects are identical).

(3) identifying a non-leaf class in the current policy tree as identical to a non-leaf class in the new policy tree upon determining that the non-leaf class in the current policy tree and the non-leaf class in the new policy tree share a greatest number of equivalent descendant leaf classes (see c. 12, ll. 39-52; Simmonds discloses synchronizing resources from a replica if a change has occurred; it is inherent that if a change has not occurred then there is no synchronization and the resources are identical; a non-leaf object is a resource); and

(4) marking the classes of the current policy tree and the new policy tree as added, deleted, modified or unchanged based on the identifying of the identical leaf and non-leaf classes in the current policy tree and new policy tree (see c. 12, ll. 39-58 and table 1 in column 13; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include unchanged, changed [or modified] and deleted; changes are marked on the resources [classes]); and

iii. selectively deleting classes of the current policy tree based on the comparison of the classes (see c. 7, l. 66 to c. 8 ll. 1-5 and c. 10, ll. 40-49; Simmonds discloses replicating and propagating changes made on the network resource tree to the replication database and selecting objects to

delete based on the comparison; the replication database is the current tree, and replicating and propagating changes involves first comparing).

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claims 2-22 and 24-43 are rejected under 35 U.S.C. 103(a) as being unpatentable over Simmonds et al. U.S. Patent No. 5,893,116 in view of Jorgensen U.S. Patent Application Publication No. 2002/0099854 A1.

13. Simmonds teaches the invention substantially as claimed including a method and machine for replicating network resources that are arranged as a hierarchical tree (see abstract and fig. 1).

14. As per claim 2, Simmonds teaches the method, as claimed in Claim 1.

Simmonds fails to teach the claimed limitation wherein the QoS packets includes a set of parameters which describe required traffic characteristics of a data connection of the packets, including a minimum bandwidth, a maximum delay, a maximum loss and jitter of the data connection, wherein each of packets includes a packet header having a

type of service field to store a value indicating a level of the QoS required for the respective packet, and wherein the level of the QoS is used to identify a class of policy for processing the respective packet by the network element.

However Jorgensen teaches QoS as related to latency, bandwidth and jitter (see paras. 0068, 0071, and 0088), analyzing IP packet headers with type of service fields to classify them into a class of service (see paras. 0119 and 0320), and a QoS matrix used to differentiate traffic into classes of service (see para. 0092).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add quality of service parameters to solve the problems of quality delivery and meet the expected level of service.

15. As per claim 3, Simmonds teaches the method, as claimed in Claim 2, wherein the leaf classes do not have a child class and are orthogonal to a remainder of the leaf classes, and wherein each non-leaf class as a parent class includes at least one leaf class as a child class and each leaf class includes a set of rules that are constrained by a parent class associated with the respective leaf class (see c. 9 l. 45-67; Simmonds discloses container objects and leaf objects, leaf objects cannot contain other objects, files are leaf objects and are perpendicular to other files, containers are non-leaf objects as they can contain other objects, leaf objects inherit rules that are applied to the preceding container object).

Simmonds fails to teach wherein each class includes a class name, a type of service, and an amount of bandwidth associated with the respective class. However, Jorgensen teaches mapping data traffic to a QoS matrix which includes class names in

order to classify data according to service type and grouping all traffic types to receive similar allocation of system and media resources. Amount of bandwidth is a system and media resource (see paras. 0091 and 0092).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add class definitions in order to implement an adequate and practical QoS mechanism to allocate different level of system resources.

16. As per clam 4, Simmonds teaches the method, as claimed in Claim 3, wherein selectively deleting classes of the current policy tree comprises deleting a class of the current policy tree upon determining that a set of classification rules of the class of the current policy tree is different than a set of classification rules of a corresponding class of the new policy tree (see c. 13, ll. 46-50 and c. 14, ll. 1-9; Simmonds discloses clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master, overwriting can involve deleting objects with different inheritance policies [classification rules]).

17. As per claim 5, Simmonds teaches the method, as claimed in Claim 4, wherein each class in the current and new policy tree is positioned at a level in the current and new policy tree and wherein the selectively deleting classes of the current policy tree comprises deleting a leaf class of the current policy tree upon determining that that the leaf class of the current policy tree is not positioned at a same level as a leaf class of the new policy tree (see c. 13, ll. 46-50 and c. 14, ll. 1-9; Simmonds discloses clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master, overwriting can involve deleting an object it does not follow

the same tree structure of the master; following the same tree structure involves being at the same level).

18. As per claim 6, Simmonds teaches the method, as claimed in Claim 3, wherein the selectively deleting classes of the current policy tree comprises selectively deleting at least one leaf class of the current policy tree (see c. 13, ll. 46-50 and c. 14, ll. 1-9; Simmonds discloses clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; the replica has leaf objects).

19. As per claim 7, Simmonds the method, as claimed in Claim 3, wherein the selectively deleting classes of the current policy tree comprises selectively deleting at least one non-leaf class of the current policy tree (see c. 13, ll. 46-50 and c. 14, ll. 1-9; Simmonds discloses clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; the replica has non-leaf objects).

20. As per claim 8, Simmonds the method, as claimed in Claim 3, wherein a class having a parent, further includes all classification rules included in the parent class, wherein a leaf class as a child of the parent class includes a set of its own rules and attributes and inherits all the rule and attributes of its parent class except a root of the respective policy tree; and wherein the rules and attributes of a child class further limit the rules and attributes of its parent class (see c. 9, ll. 60 to c. 10, ll. 27; Simmonds discloses leaf objects inherit inheritance rules from the container and containers inherit inheritance rules from its parent container; inheritance rules are classification rules. Simmonds also discloses inheritance rules apply to leaf classes from the container

objects up to the root. The root attributes contained in the root are not inherited by leaf classes. Leaf rules and attributes apply to the parent container).

Simmonds fails to teach the root representing a data link associated with an output port of the network element. However, Jorgensen teaches the data link layer includes a media access control (MAC) layer. The MAC represents a unique network element identifier with a port to the network (see para. 0317).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add the data link layer to provide an output to the network of the analyzed and scheduled classes.

21. As per claim 9, Simmonds the method, as claimed in Claim 3, wherein each class is positioned at a level in a policy tree and wherein a leaf class of the current policy tree is identical to a leaf class of the new policy tree only if the leaf class of the current policy tree and the leaf class of the new policy tree are positioned at an equal level (see c. 12, ll. 39-58 and table 1 in column 13; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include unchanged; if a leaf is unchanged then it is positioned at the same level on both trees and thus, identical).

22. As per claim 10, Simmonds the method, as claimed in Claim 9, wherein each leaf class in the current policy tree and the new policy tree is reciprocally linked to an associated path of non-leaf classes in the current policy tree and new policy tree, respectively, and wherein the selectively deleting the classes of the current policy tree comprises deleting each leaf class in the current policy tree upon determining that the

associated path of non-leaf classes in the current policy tree is different from the path of non-leaf classes in the new policy tree for a leaf class (see c. 12, ll. 39-58 and table 1 and c. 14, ll. 1-9; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include missing directories [or parents] (non-leaf classes), and clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; overwriting can involve deleting leaf objects without parents).

23. As per claim 11, Simmonds the method, as claimed in Claim 9, wherein each class in the current and new policy tree is positioned at a level in the current and new policy tree, wherein each leaf class in the current policy tree and the new policy tree is reciprocally linked to an associated path of non-leaf classes in the current policy tree and new policy tree, respectively, and wherein the selectively deleting classes of the current policy tree comprises deleting a leaf class of the current policy tree upon determining that the associated path of non-leaf classes linked to the leaf class of the current policy tree includes a non-leaf class positioned at a different level than a corresponding non-leaf class included in the associated path of non-leaf classes linked to the leaf class of the new policy tree (see c. 12, ll. 39-58 and table 1 and c. 14, ll. 1-9; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include missing directories [or parents] (or non-leaf classes), and clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; overwriting can involve deleting leaf objects with different parents at different levels).

24. As per claim 12, Simmonds the method, as claimed in Claim 11, selectively deleting classes of the current policy tree comprises deleting a leaf class of the current policy tree upon determining that all ancestors of the leaf class of the current policy tree and corresponding ancestors of the leaf of the new policy tree have fewer identical descendant classes than those had by a class of the current policy tree and a class of the new policy tree positioned at the same level as the parents of the leaf class of the current policy tree and the ancestors of the leaf class of the new policy tree (see c. 12, ll. 39-58, table 1 in column 13 and c. 14, ll. 1-9; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include changed; if a leaf is changed then it is not identical resulting in a clash and clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; overwriting can involve deleting leaf objects if they are not identical).

25. As per claim 13, Simmonds teaches the method, as claimed in Claim 8.

Simmonds fails to teach the claimed limitation wherein each of the classes having at least two children is identified as a scheduling class and each of the classes not having a child class is identified as a flow class, wherein packets of each flow class are processed in an order of the packets stored in a queue and packets of each scheduling class are processed according to a predetermined schedule and wherein at least one of the children classes of a scheduling class contains one or more flows that match the respective scheduling class and are not contained in a remainder of the children classes associated with the respective scheduling class.

However, Jorgensen teaches hierarchical class-based queuing classifying different types of IP flows using a tree structure at the edge access devices, each branch signifies a different flow and scheduling each subframe according to priorities (see paras. 0506 0317, and 0507).

It would have been obvious to one of ordinary skill in the art at the time of the invention class-based queuing and scheduling in order to move the packets in accordance to the class they belong to in compliance with QoS and service level agreements.

26. As per claim 14, Simmonds teaches the method, as claimed in Claim 13.

Simmonds fails to teach the claimed limitation further comprising:

a. verifying each class of the new policy tree with respect to remaining classes of the new policy tree to avoid conflicts when the new policy tree is merged into a merged policy tree;

b. associating the merged policy tree with a termination point to activate the merged policy tree on the termination point; and

c. processing data packets using a packet processing pipeline according to the merged policy tree.

However, Jorgensen teaches differentiating traffic with respect to class to simplify operation of QoS mechanism (see para. 0092), the processor module takes packets queued in class queues, and scheduling packets in queues according to priorities (see para. 0567).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add traffic differentiating and scheduling in order to move the packets in accordance to the class they belong to in compliance with QoS and service level agreements.

27. As per claim 15, Simmonds teaches the method, as claimed in Claim 14.

Simmonds fails to teach the claimed limitation wherein the packet processing pipeline comprises:

a. and incoming packet manager (IPM) to examine a packet header of data packets to determine a next hope of the data packets;

b. a class identifier (CI) coupled to the IPM to classify the data packets using the merged policy tree;

c. a route identifier (RI) coupled to the CI to determine which output port through which each of the data packets should be routed;

d. an outgoing packet manager (OPM) coupled to the RI to store the data packets for outgoing purposes;

e. a flow identifier (FI) coupled to the OPM to identify one or more flows of which the data packets belong; and

f. a traffic manager coupled to the FI to schedule the data packets out of the output port using a result of the FI and the merged policy tree.

However, Jorgensen teaches an IP flow analyzer receives and analyzes IP flow according to packet header fields with destination data, classifies IP flow and group together IP flows with similar QoS requirements (para. 0371), and an IP flow scheduler

schedules received IP flows and transmits them to a downlink (para. 0372). Scheduling packets involves storing packets.

It would have been obvious to one of ordinary skill in the art at the time of the invention to add an IP flow analyzer and scheduler in order to move the packets in accordance to the class they belong to in compliance with QoS and service level agreements.

28. As per claim 16, Simmonds teaches the method, as claimed in Claim 15.

Simmonds fails to teach the claimed limitation wherein the network element includes an Ethernet interface card (EIC) coupled to a local area network (LAN) and an ATM interface card (AIC) coupled to a wide area network (WAN), and wherein processing data packets using a packet processing pipeline further comprises:

a. in response to a data packet received at the EIC from the LAN, a CI of the EIC classifying the data packet using identification of the policy tree;

b. an RI of the EIC determining an output port through which the data packet should be routed using information of the policy tree;

c. a FI of the AIC determining a flow to which the data packet belongs, using the policy tree; and

d. transmitting the data packet to the WAN through the output port according to the QoS based on the policy tree.

However, Jorgensen teaches a data network can act as a wide area network (WAN) for coupling a plurality of local area networks (LANs) together by network interface cards. The networks can be Ethernet switch or ATM networks (see paras.

0208, 0218, and 0219). Ethernet interface cards must be used in Ethernet switch networks and ATM interface cards must be used in ATM networks. Jorgensen also teaches an IP flow analyzer receives and analyzes IP flow according to packet header fields with destination data, classifies IP flows and groups together IP flows with similar QoS requirements (para. 0371), and an IP flow scheduler schedules received IP flows and transmits them to a downlink (para. 0372). Scheduling packets involves storing packets.

It would have been obvious to one of ordinary skill in the art at the time of the invention to add network interface cards to facilitate the IP flow analyzer and scheduler in moving packets in accordance to the class they belong to in compliance with QoS and service level agreements.

29. As per claim 17, Simmonds teaches the method, as claimed in Claim 16.

Simmonds fails to teach the claimed limitation wherein the network element further comprises a controller card, wherein the controller card is to perform following operations:

a. compiling each policy tree and generate a class lookup table (CLT) accessible by a CI of the EIC;

b. associating each policy tree with a termination point and generate a routing table accessible by each RI for looking up a next hop baed on a class ID; and

c. creating a list of flow identifier and scheduling update commands to incrementally change the flow of tables for FIs and traffic manager tables to TMs using flow class and scheduling class property information of the policy tree.

However, Jorgensen teaches a module in which a packet is classified into a QoS class by performing a table lookup into another IP flow class table module which stores the types of QoS classes (see para. 0489). Jorgensen also teaches a router as a network interface. As such, a router acts as an interface between two networks, routing a packet based on its destination address (see paras. 0215 and 0216). Jorgensen also teaches a process which consults the local routing database to obtain routes and the QoS of those packets as they are forwarded in accordance with that routing. Routing protocols determine where packets are forwarded, thus updating commands as classes change (see para. 0342). Packets are identified at each router hop as either low or high priority (see para. 0326).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add a physical router to help facilitate the moving packets in accordance to the class they belong to in compliance with QoS and service level agreements.

30. As per claim 18, Simmonds teaches the method, as claimed in Claim 17.

Simmonds fails to teach the claimed limitation wherein associating the policy tree with a termination point comprises:

a. creating tables required by the CI and OPM;

b. differentiating between classes that are currently in service from classes that will be put into services to generate a list of FI and TM update commands;

c. distributing and synchronizing deleted classes by applying the tables and delete commands;

d. distributing and synchronizing added classes by applying the tables and add commands; and

e. distributing and synchronizing modified classes by applying the tables and modify commands.

However, Jorgensen teaches a module in which a packet is classified into a QoS class by performing a table lookup into another IP flow class table module which stores the types of QoS classes (see para. 0489), existing and new IP flow class queuing - existing IP flow classes are current classes and new IP flow classes are classes that will be put into service (see paras. 0492 and 0493), and queuing changes based on revision of administrative information regarding QoS and service level agreement priority. Queuing is based on the class, if the class changes then the queuing changes. Changes in classes include deletion, addition and modification (see paras. 0502 and 0504).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add class table lookup for IP flows in order to move packets in accordance to the class they belong to in compliance with QoS and service level agreements.

31. As per claim 19, Simmonds teaches the method, as claimed in Claim 18.

Simmonds fails to teach the limitation as claimed wherein each class identifies a subset of packets using one or more classification rules, each classification rule including one or more rule terms, and each rule term including an identity data item and a set of constant values associates with the data item, wherein the set of constant values includes at least one of individual values ranges of constant values, IP subnets

expressed in a notation of A.B.C.D/E where A.B.C.D is an IP address and /E indicates a number of leading bits that identify the subnet portion of the IP address.

However, Jorgensen teaches the flow scheduler places the data packets of an IP flow into a class queue based on class queue priorities and transmits them using a set of rules. The rules are determined by inputs based on hierarchical class-based prioritization, directory-enabled data priority and service level agreement priority (see para. 0385).

"Official Notice" is taken that the A.B.C.D/E notation for indicating the subnet of a given IP address in the form of A.B.C.D is old and well known in the art and is known as Classless InterDomain Routing (CIDR).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add priorities and rules to queuing and scheduling classes in order to facilitate the moving of packets in accordance to the class they belong to in compliance with QoS and service level agreements.

32. As per claim 20, Simmonds teaches the method, as claimed in Claim 19.

Simmonds fails to teach the limitation as claimed wherein the data item is one of a source IP address, destination IP address, TCP/UDP source port, TCP/UDP destination port, ingress port of the network element, a type of service byte, a protocol type, and TCP acknowledgement flag.

However, Jorgensen teaches data traffic differentiated by information contained in packet headers such as source IP address, source TCP or UDP port, destination address, destination IP or UDP port, IP type of service. A ping (echo packet) serves as

an acknowledgement flag. The address resolution protocol maps the physical address to an IP address, and thus the ingress port of the network element is known (see paras. 0119 and 0320).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add data traffic differentiation based on packet-contained information in order to facilitate the moving packets in accordance to the class they belong to in compliance with QoS and service level agreements.

33. As per claim 21, Simmonds teaches the method, as claimed in Claim 20.

Simmonds fails to teach the claimed limitation wherein when a data packet is received at the network element, classification rules are evaluated to classify the data packet, wherein if the value of the data item matched any of the constant values associated with a rule term, the rule term is satisfied, wherein the rule is satisfied, the data packet is considered to belong to a class associated with the satisfied classification rule.

However, Jorgensen teaches an IP flow analyzer receives and analyzes IP flow according to packet header fields with destination data, classifies the IP flow and groups together IP flows with similar QoS requirements (paras. 0371 and 0383).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add an IP flow analyzer in order to classify packets in order to move them in accordance to the class they belong to in compliance with QoS and service level agreements.

34. As per claim 22, Simmonds teaches the method, as claimed in Claim 18.

Simmonds fails to teach the claimed limitation wherein each class comprises one or more classification rules, QoS requirements, and a classification mask to specify which dimensions are specified in terms of the one or more classification rules, where the classification mask comprises:

a. bit 0 to indicate a source IP address;

b. bit 1 to indicate a destination IP address;

c. bit 2 to indicate a source TCP/UDP port;

d. bit 3 to indicate a destination TCP/UDP port;

e. bit 4 to indicate an incoming port;

f. bit 5 to indicate a type of service byte;

g. bit 6 to indicate a type of protocol used; and

h. bit 7 to indicate a TCP ACK flag

However, Jorgensen teaches data packets are placed into priority class queues then transmitted based on rules, the header information used to identify QoS requirements consists of as source IP address, source TCP or UDP port, destination address, destination IP or UDP port, IP type of service. A ping (echo packet) serves as an acknowledgement flag. The address resolution protocol maps the physical address to an IP address, and thus the incoming port is known (see paras. 0119, 0320, 0385, 0391, 0406 to 0410).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add rules and QoS requirements in classification of packets in order to

move them in accordance to the class they belong to in compliance with QoS and service level agreements.

35. As per claim 24, Simmonds teaches the machine-readable medium, as claimed in Claim 23.

Simmonds fails to teach the claimed limitation wherein the QoS packets includes a set of parameters which describe required traffic characteristics of a data connection of the packets, including a minimum bandwidth, a maximum delay, a maximum loss and jitter of the data connection, wherein each of packets includes a packet header having a type of service field to store a value indicating a level of the QoS required for the respective packet, and wherein the level of the QoS is used to identify a class of policy for processing the respective packet by the network element.

However Jorgensen teaches QoS as related to latency, bandwidth and jitter (see paras. 0068, 0071, and 0088), analyzing IP packet headers with type of service fields to classify them into a class of service (see paras. 0119 and 0320), and a QoS matrix used to differentiate traffic into classes of service (see para. 0092).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add quality of service parameters to solve the problems of quality delivery and meet the expected level of service.

36. As per claim 25, Simmonds teaches the machine-readable medium, as claimed in Claim 24, wherein the leaf classes do not have a child class and are orthogonal to a remainder of the leaf classes, and wherein each non-leaf class as a parent class includes at least one leaf class as a child class and each leaf class includes a set of

rules that are constrained by a parent class associated with the respective leaf class (see c. 9 l. 45-67; Simmonds discloses container objects and leaf objects, leaf objects cannot contain other objects, files are leaf objects and are perpendicular to other files, containers are non-leaf objects as they can contain other objects, leaf objects inherit rules that are applied to the preceding container object).

Simmonds fails to teach wherein each class includes a class name, a type of service, and an amount of bandwidth associated with the respective class. However, Jorgensen teaches mapping data traffic to a QoS matrix which includes class names in order to classify data according to service type and grouping all traffic types to receive similar allocation of system and media resources. Amount of bandwidth is a system and media resource (see paras. 0091 and 0092).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add class definitions in order to implement an adequate and practical QoS mechanism to allocate different level of system resources.

37. As per claim 26, Simmonds teaches the machine-readable medium, as claimed in Claim 25, wherein selectively deleting classes of the current policy tree comprises deleting a class of the current policy tree upon determining that a set of classification rules of the class of the current policy tree is different than a set of classification rules of a corresponding class of the new policy tree (see c. 13, ll. 46-50 and c. 14, ll. 1-9; Simmonds discloses clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master, overwriting can involve deleting objects with different inheritance policies [classification rules]).

38. As per claim 27, Simmonds teaches the machine-readable medium, as claimed in Claim 26, wherein each class in the current and new policy tree is positioned at a level in the current and new policy tree and wherein the selectively deleting classes of the current policy tree comprises deleting a leaf class of the current policy tree upon determining that that the leaf class of the current policy tree is not positioned at a same level as a leaf class of the new policy tree (see c. 13, ll. 46-50 and c. 14, ll. 1-9; Simmonds discloses clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master, overwriting can involve deleting an object it does not follow the same tree structure of the master; following the same tree structure involves being at the same level).

39. As per claim 28, Simmonds teaches the machine-readable medium, as claimed in Claim 25, wherein the selectively deleting classes of the current policy tree comprises selectively deleting at least one leaf class of the current policy tree (see c. 13, ll. 46-50 and c. 14, ll. 1-9; Simmonds discloses clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; the replica has leaf objects).

40. As per claim 29, Simmonds teaches the machine-readable medium, as claimed in Claim 25, wherein the selectively deleting classes of the current policy tree comprises selectively deleting at least one non-leaf class of the current policy tree (see c. 13, ll. 46-50 and c. 14, ll. 1-9; Simmonds discloses clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; the replica has non-leaf objects).

41. As per claim 30, Simmonds teaches the machine-readable medium, as claimed in Claim 25, wherein a class having a parent, further includes all classification rules included in the parent class, wherein a leaf class as a child of the parent class includes a set of its own rules and attributes and inherits all the rule and attributes of its parent class except a root of the respective policy tree; and wherein the rules and attributes of a child class further limit the rules and attributes of its parent class (see c. 9, ll. 60 to c. 10, ll. 27; Simmonds discloses leaf objects inherit inheritance rules from the container and containers inherit inheritance rules from its parent container; inheritance rules are classification rules. Simmonds also discloses inheritance rules apply to leaf classes from the container objects up to the root. The root attributes contained in the root are not inherited by leaf classes. Leaf rules and attributes apply to the parent container).

Simmonds fails to teach the root representing a data link associated with an output port of the network element. However, Jorgensen teaches the data link layer includes a media access control (MAC) layer. The MAC represents a unique network element identifier with a port to the network (see para. 0317).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add the data link layer to provide an output to the network of the analyzed and scheduled classes.

42. As per claim 31, Simmonds teaches the machine-readable medium, as claimed in Claim 25, wherein each class is positioned at a level in a policy tree and wherein a leaf class of the current policy tree is identical to a leaf class of the new policy tree only if the leaf class of the current policy tree and the leaf class of the new policy tree are

positioned at an equal level (see c. 12, ll. 39-58 and table 1 in column 13; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include unchanged; if a leaf is unchanged then it is positioned at the same level on both trees and thus, identical).

43. As per claim 32, Simmonds teaches the machine-readable medium, as claimed in Claim 31, wherein each leaf class in the current policy tree and the new policy tree is reciprocally linked to an associated path of non-leaf classes in the current policy tree and new policy tree, respectively, and wherein the selectively deleting the classes of the current policy tree comprises deleting each leaf class in the current policy tree upon determining that the associated path of non-leaf classes in the current policy tree is different from the path of non-leaf classes in the new policy tree for a leaf class (see c. 12, ll. 39-58 and table 1 and c. 14, ll. 1-9; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include missing directories [or parents] (non-leaf classes), and clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; overwriting can involve deleting leaf objects without parents).

44. As per claim 33, Simmonds teaches the machine-readable medium, as claimed in Claim 25, wherein each class in the current and new policy tree is positioned at a level in the current and new policy tree, wherein each leaf class in the current policy tree and the new policy tree is reciprocally linked to an associated path of non-leaf classes in the current policy tree and new policy tree, respectively, and wherein the selectively deleting classes of the current policy tree comprises deleting a leaf class of the current

policy tree upon determining that the associated path of non-leaf classes linked to the leaf class of the current policy tree includes a non-leaf class positioned at a different level than a corresponding non-leaf class included in the associated path of non-leaf classes linked to the leaf class of the new policy tree (see c. 12, ll. 39-58 and table 1 and c. 14, ll. 1-9; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include missing directories [or parents] (or non-leaf classes), and clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; overwriting can involve deleting leaf objects with different parents at different levels).

45. As per claim 34, machine-readable medium, as claimed in Claim 33, wherein the selectively deleting classes of the current policy tree comprises deleting a leaf class of the current policy tree upon determining that all parents of the leaf class of the current policy tree and corresponding parents of the leaf of the new policy tree have fewer identical descendant classes than those had by a class of the current policy tree and a class of the new policy tree positioned at the same level as the parents of the leaf class of the current policy tree and the parents of the leaf class of the new policy tree (see c. 12, ll. 39-58, table 1 in column 13 and c. 14, ll. 1-9; Simmonds discloses changes in resources in both the replica and the network resource tree, possible change types include changed; if a leaf is changed then it is not identical resulting in a clash and clash handling of incompatible changes in the replica and master are resolved by overwriting the replica with the master; overwriting can involve deleting leaf objects if they are not identical).

46. As per claim 35, Simmonds teaches the machine-readable medium, as claimed in Claim 30.

Simmonds fails to teach the claimed limitation wherein each of the classes having at least two children is identified as a scheduling class and each of the classes not having a child class is identified a flow class, wherein packets of each flow class are processed in an order of the packets stored in a queue and packets of each scheduling class are processed according to a predetermined schedule and wherein at least one of the children classes of a scheduling class contains one or more flows that match the respective scheduling class and are not contained in a remainder of the children classes associated with the respective scheduling class.

However, Jorgensen teaches hierarchical class-based queuing classifying different types of IP flows using a tree structure at the edge access devices, each branch signifies a different flow and scheduling each subframe according to priorities (see paras. 0506 0317, and 0507).

It would have been obvious to one of ordinary skill in the art at the time of the invention class-based queuing and scheduling in order to move the packets in accordance to the class they belong to in compliance with QoS and service level agreements.

47. As per claim 36, Simmonds teaches the machine-readable medium, as claimed in Claim 35.

Simmonds fails to teach the claimed limitation further comprising:

verifying each class of the new policy tree with respect to remaining classes of the new policy tree to avoid conflicts when the new policy tree is merged into a merged policy tree;

associating the merged policy tree with a termination point to activate the merged policy tree on the termination point; and

processing data packets using a packet processing pipeline according to the merged policy tree.

However, Jorgensen teaches differentiating traffic with respect to class to simplify operation of QoS mechanism (see para. 0092), the processor module takes packets queued in class queues, and scheduling packets in queues according to priorities (see para. 0567).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add traffic differentiating and scheduling in order to move the packets in accordance to the class they belong to in compliance with QoS and service level agreements.

48. As per claim 37, Simmonds teaches the machine-readable medium, as claimed in claim 36.

Simmonds fails to teach the claimed limitation wherein the packet processing pipeline comprises:

a. and incoming packet manager (IPM) to examine a packet header of data packets to determine a next hope of the data packets;

b. a class identifier (CI) coupled to the IPM to classify the data packets using the merged policy tree;

c. a route identifier (RI) coupled to the CI to determine which output port through which each of the data packets should be routed;

d. an outgoing packet manager (OPM) coupled to the RI to store the data packets for outgoing purposes;

e. a flow identifier (FI) coupled to the OPM to identify one or more flows of which the data packets belong; and

f. a traffic manager coupled to the FI to schedule the data packets out of the output port using a result of the FI and the merged policy tree.

However, Jorgensen teaches an IP flow analyzer receives and analyzes IP flow according to packet header fields with destination data, classifies IP flow and group together IP flows with similar QoS requirements (para. 0371), and an IP flow scheduler schedules received IP flows and transmits them to a downlink (para. 0372). Scheduling packets involves storing packets.

It would have been obvious to one of ordinary skill in the art at the time of the invention to add an IP flow analyzer and scheduler in order to move the packets in accordance to the class they belong to in compliance with QoS and service level agreements.

49. As per claim 38, Simmonds teaches the machine-readable medium, as claimed in Claim 37.

Simmonds fails to teach the claimed limitation wherein the network element includes an Ethernet interface card (EIC) coupled to a local area network (LAN) and an ATM interface card (AIC) coupled to a wide area network (WAN), and wherein processing data packets using a packet processing pipeline further comprises:

a. in response to a data packet received at the EIC from the LAN, a CI of the EIC classifying the data packet using identification of the policy tree;

b. an RI of the EIC determining an output port through which the data packet should be routed using information of the policy tree;

c. a FI of the AIC determining a flow to which the data packet belongs, using the policy tree; and

d. transmitting the data packet to the WAN through the output port according to the QoS based on the policy tree.

However, Jorgensen teaches a data network can act as a wide area network (WAN) for coupling a plurality of local area networks (LANs) together by network interface cards. The networks can be Ethernet switch or ATM networks (see paras. 0208, 0218, and 0219). Ethernet interface cards must be used in Ethernet switch networks and ATM interface cards must be used in ATM networks. Jorgensen also teaches an IP flow analyzer receives and analyzes IP flow according to packet header fields with destination data, classifies IP flows and groups together IP flows with similar QoS requirements (para. 0371), and an IP flow scheduler schedules received IP flows and transmits them to a downlink (para. 0372). Scheduling packets involves storing packets.

It would have been obvious to one of ordinary skill in the art at the time of the invention to add network interface cards to facilitate the IP flow analyzer and scheduler in moving packets in accordance to the class they belong to in compliance with QoS and service level agreements.

50. As per claim 39, Simmonds teaches the machine-readable medium, as claimed in Claim 38.

Simmonds fails to teach the claimed limitation wherein the network element further comprises a controller card, wherein the controller card is to perform following operations:

a. compiling each policy tree and generate a class lookup table (CLT) accessible by a CI of the EIC;

b. associating each policy tree with a termination point and generate a routing table accessible by each RI for looking up a next hop baed on a class ID; and

c. creating a list of flow identifier and scheduling update commands to incrementally change the flow of tables for FIs and traffic manager tables to TMs using flow class and scheduling class property information of the policy tree.

However, Jorgensen teaches a module in which a packet is classified into a QoS class by performing a table lookup into another IP flow class table module which stores the types of QoS classes (see para. 0489). Jorgensen also teaches a router as a network interface. As such, a router acts as an interface between two networks, routing a packet based on its destination address (see paras. 0215 and 0216). Jorgensen also teaches a process which consults the local routing database to obtain

routes and the QoS of those packets as they are forwarded in accordance with that routing. Routing protocols determine where packets are forwarded, thus updating commands as classes change (see para. 0342). Packets are identified at each router hop as either low or high priority (see para. 0326).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add a physical router to help facilitate the moving packets in accordance to the class they belong to in compliance with QoS and service level agreements.

51. As per claim 40, Simmonds teaches the machine-readable medium, as claimed in Claim 39.

Simmonds fails to teach the claimed limitation wherein associating the policy tree with a termination point comprises:

a. creating tables required by the CI and OPM;

b. differentiating between classes that are currently in service from classes that will be put into services to generate a list of FI and TM update commands;

c. distributing and synchronizing deleted classes by applying the tables and delete commands;

d. distributing and synchronizing deleted classes by applying the tables and add commands; and

e. distributing and synchronizing deleted classes by applying the tables and modify commands.

However, Jorgensen teaches a module in which a packet is classified into a QoS class by performing a table lookup into another IP flow class table module which stores

the types of QoS classes (see para. 0489), existing and new IP flow class queuing - existing IP flow classes are current classes and new IP flow classes are classes that will be put into service (see paras. 0492 and 0493), and queuing changes based on revision of administrative information regarding QoS and service level agreement priority. Queuing is based on the class, if the class changes then the queuing changes. Changes in classes include deletion, addition and modification (see paras. 0502 and 0504).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add class table lookup for IP flows in order to move packets in accordance to the class they belong to in compliance with QoS and service level agreements.

52. As per claim 41, Simmonds teaches the machine-readable medium, as claimed in Claim 40.

Simmonds fails to teach the limitation as claimed wherein each class identifies a subset of packets using one or more classification rules, each classification rule including one or more rule terms, and each rule term including an identity data item and a set of constant values associates with the data item, wherein the set of constant values includes at least one of individual values ranges of constant values, IP subnets expressed in a notation of A.B.C.D/E where A.B.C.D is an IP address and /E indicates a number of leading bits that identify the subnet portion of the IP address.

However, Jorgensen teaches the flow scheduler places the data packets of an IP flow into a class queue based on class queue priorities and transmits them using a set of rules. The rules are determined by inputs based on hierarchical class-based

prioritization, directory-enabled data priority and service level agreement priority (see para. 0385).

"Official Notice" is taken that the A.B.C.D/E notation for indicating the subnet of a given IP address in the form of A.B.C.D is old and well known in the art and is known as Classless InterDomain Routing (CIDR).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add priorities and rules to queuing and scheduling classes in order to facilitate the moving of packets in accordance to the class they belong to in compliance with QoS and service level agreements.

53. As per claim 42, Simmonds teaches the machine-readable medium, as claimed in Claim 41.

Simmonds fails to teach the limitation as claimed wherein the data item is one of a source IP address, destination IP address, TCP/UDP source port, TCP/UDP destination port, ingress port of the network element, a type of service byte, a protocol type, and TCP acknowledgement flag.

However, Jorgensen teaches data traffic differentiated by information contained in packet headers such as source IP address, source TCP or UDP port, destination address, destination IP or UDP port, IP type of service. A ping (echo packet) serves as an acknowledgement flag. The address resolution protocol maps the physical address to an IP address, and thus the ingress port of the network element is known (see paras. 0119 and 0320).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add data traffic differentiation based on packet-contained information in order to facilitate the moving packets in accordance to the class they belong to in compliance with QoS and service level agreements.

54. As per claim 43, Simmonds teaches the machine-readable medium, as claimed in Claim 42.

Simmonds fails to teach the claimed limitation wherein when a data packet is received at the network element, classification rules are evaluated to classify the data packet, wherein if the value of the data item matched any of the constant values associated with a rule term, the rule term is satisfied , wherein the rule is satisfied, the data packet is considered to belong to a class associated with the satisfied classification rule.

However, Jorgensen teaches an IP flow analyzer receives and analyzes IP flow according to packet header fields with destination data, classifies the IP flow and groups together IP flows with similar QoS requirements (paras. 0371 and 0383).

It would have been obvious to one of ordinary skill in the art at the time of the invention to add an IP flow analyzer in order to classify packets in order to move them in accordance to the class they belong to in compliance with QoS and service level agreements.

Response to Arguments

55. Applicant's arguments with respect to claims 1-44 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

56. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

57. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ciara Martin whose telephone number is 571-272-7507. The examiner can normally be reached on M-F 6:30- 4:00 with second Fridays off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ario Etienne can be reached on 571-272-4001. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

CM
4/28/05

SALEH NAJJAR
PRIMARY EXAMINER